**零声学院出品**

QQ群：音视频技术交流群 782508536

依依老师QQ：2693590767

秋香老师QQ：2207032995

音视频流媒体高级开发课程：https://ke.qq.com/course/468797?tuin=137bb271

课程简介：

AppRTC是谷歌官方提供的WebRTC开源项目，它的价值主要在于：

（1）给出信令房间的范例；

（2）给出各种浏览器的兼容方法。

对于初学者而言，重要的是先把AppRTC跑起来，但初次接触WebRTC是很难一次顺利把AppRTC跑起来的，比如：

（1）http网页打不开摄像头怎么办？

（2）https没有证书怎么办？

（3）为什么信令老是连接不成功？

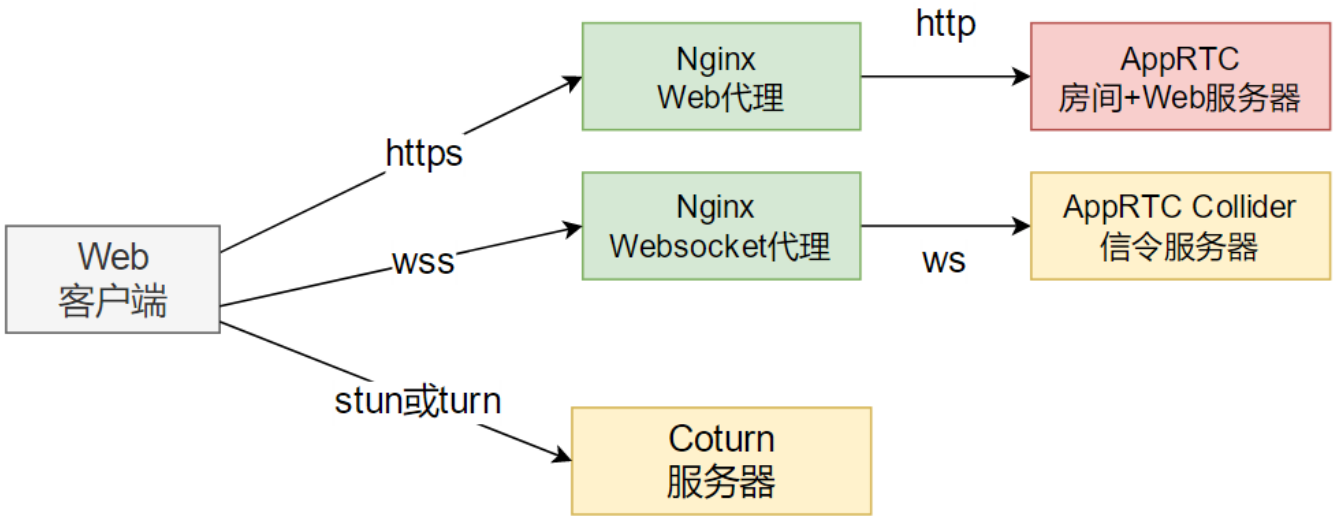
# 搭建AppRTC

**搭建环境ubuntu 16.04server版本**

## 1. 服务器组成

1. AppRTC 房间+Web服务器 https://github.com/webrtc/apprtc
2. Collider 信令服务器，在AppRTC源码里
3. CoTurn coturn打洞+中继服务器
4. Nginx 服务器，用于Web访问代理和Websocket代理。

AppRTC组成图如下所示。




AppRTC 房间+Web服务器使用python+js语言

AppRTC Collider信令服务器采用go语言

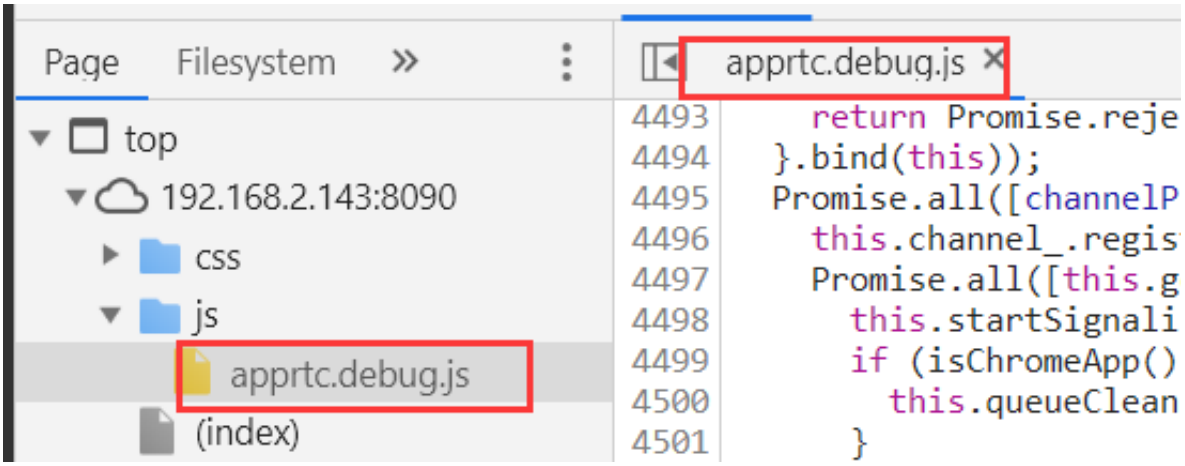
Coturn 采用C语言

在部署到公网时需要通过Nginx做Web和Websocket的代理连接

实际开发：把信令+房间管理 都是写到一个服务器

## AppRTC的的价值：

（1）js代码；apprtc/out/chrome_app/js/apprtc.debug.js




（2）Collider信令服务器原型。

# 2 准备工作

在一台全新的ubuntu 16.04 server版本安装AppRTC，前期准备工作。

1. 安装vim
2. 安装ssh
3. 安装ifconfig工具
4. 更新源
5. 安装git

## 2.1 安装vim

```
sudo apt-get install vim
```

## 2.2 安装ssh

```
sudo apt-get install openssh-server
```

输入 "sudo ps -e | grep ssh" --> 回车 --> 有 sshd，说明 ssh 服务已经启动，如果没有启动，输入 "sudo service ssh start" --> 回车 --> ssh 服务就会启动。

## 2.3 安装ifconfig工具

```
sudo apt-get install net-tools
sudo apt-get install iputils-ping
```

## 2.4 更新源

将源更新为阿里源，否则apt-get install安装软件较慢。

```
# 1 在修改source.list前，最好先备份一份
sudo mv /etc/apt/sources.list /etc/apt/sources.list.old
# 2 执行命令打开sourcse.list文件
sudo vim /etc/apt/sources.list
# 3 复制更新源
```

复制以下源到sources.list

```
# deb cdrom:[Ubuntu 16.04 LTS _Xenial Xerus_ - Release amd64 (20160420.1)]/ xenial main restricted
deb-src http://archive.ubuntu.com/ubuntu xenial main restricted #Added by software-properties
deb http://mirrors.aliyun.com/ubuntu/ xenial main restricted
deb-src http://mirrors.aliyun.com/ubuntu/ xenial main restricted multiverse universe #Added by software-properties
deb http://mirrors.aliyun.com/ubuntu/ xenial-updates main restricted
deb-src http://mirrors.aliyun.com/ubuntu/ xenial-updates main restricted multiverse universe #Added by software-properties
deb http://mirrors.aliyun.com/ubuntu/ xenial universe
```

```
deb http://mirrors.aliyun.com/ubuntu/ xenial-updates universe
deb http://mirrors.aliyun.com/ubuntu/ xenial multiverse
deb http://mirrors.aliyun.com/ubuntu/ xenial-updates multiverse
deb http://mirrors.aliyun.com/ubuntu/ xenial-backports main restricted universe multiverse
deb-src http://mirrors.aliyun.com/ubuntu/ xenial-backports main restricted universe multiverse #Added by software-properties
deb http://archive.canonical.com/ubuntu xenial partner
deb-src http://archive.canonical.com/ubuntu xenial partner
deb http://mirrors.aliyun.com/ubuntu/ xenial-security main restricted
deb-src http://mirrors.aliyun.com/ubuntu/ xenial-security main restricted multiverse universe #Added by software-properties
deb http://mirrors.aliyun.com/ubuntu/ xenial-security universe
deb http://mirrors.aliyun.com/ubuntu/ xenial-security multiverse
```

保存后更新源

```
# 4 update命令
sudo apt-get update
```

## 2.5 安装git

```
sudo apt-get install git
```

# 3 安装AppRTC必须的软件

## 3.0 创建目录

```
mkdir ~/webrtc
cd ~/webrtc
```

注意后续的目录，因为我的webrtc全路径为**/home/lqf/webrtc**，所以后续的目录都是采用该目录。

安装需要的各种工具(除了apt之外还可以下载安装包或者源码自己编译安装)：

## 3.1 安装JDK

```
# 先安装add-apt-repository命令支持
sudo apt-get install python-software-properties
sudo apt-get install software-properties-common
# 第一步：打开终端，添加ppa源
sudo add-apt-repository ppa:openjdk-r/ppa
# 第二步：更新源
sudo apt-get update
# 第三步：安装openjdk 8
sudo apt-get install openjdk-8-jdk
# 第四步：配置openjdk 8为默认java环境
sudo update-alternatives --config java
sudo update-alternatives --config javac
# 最后，验证一下是否成功
java -version
# 返回
openjdk version "1.8.0_222"
OpenJDK Runtime Environment (build 1.8.0_222-8u222-b10-1ubuntu1~16.04.1-b10)
OpenJDK 64-Bit Server VM (build 25.222-b10, mixed mode)
```

## 3.2 安装node.js

```
cd ~/webrtc
wget https://nodejs.org/dist/v10.16.0/node-v10.16.0-linux-x64.tar.xz
# 解压
tar -xvf node-v10.16.0-linux-x64.tar.xz
# 进入目录
cd node-v10.16.0-linux-x64/
# 查看当前的目录
```

```
pwd

# 确认一下nodejs下bin目录是否有node 和npm文件，如果有就可以执行软连接，比如
sudo ln -s /home/lqf/webrtc/node-v10.16.0-linux-x64/bin/npm /usr/local/bin/
sudo ln -s /home/lqf/webrtc/node-v10.16.0-linux-x64/bin/node /usr/local/bin/

# 看清楚，这个路径是你自己创建的路径，我的路径是/home/lqf/webrtc/node-v10.16.0-linux-x64

# 查看是否安装，安装正常则打印版本号
node -v
npm -v


# 有版本信息后安装 grunt-cli,先进到nodejs的bin目录，要和自己的目录匹配
cd /home/lqf/webrtc/node-v10.16.0-linux-x64/bin
sudo npm -g install grunt-cli
sudo ln -s /home/lqf/webrtc/node-v10.16.0-linux-x64/bin/grunt /usr/local/bin/

grunt --version
# 显示grunt-cli v1.3.2

#使用淘宝源替换npm，后续要执行npm时执行cnpm
sudo npm install -g cnpm --registry=https://registry.npm.taobao.org
sudo ln -s /home/lqf/webrtc/node-v10.16.0-linux-x64/lib/node_modules/cnpm/bin/cnpm /usr/local/bin/
```

## 3.3 安装Python和Python-webtest （python2.7）

ubuntu 一般都会自带python 2.7 输入 python -v 输出版本则已经有了

```
lqf@ubuntu:~$ python -V
Python 2.7.12
```

如果没有则安装

```

# 没有则输入下命令
```

```
sudo apt-get install python
# python 安装之后安装 Python-webtest
sudo apt-get install python-webtest


python -V
#Python 2.x
```

## 3.4 安装google_appengine

```
# 回到webrtc目录
cd ~/webrtc/
# 下载google_appengine
wget https://storage.googleapis.com/appengine-sdks/featured/google_appengine_1.9.40.zip
unzip google_appengine_1.9.40.zip

#配置环境变量：在/etc/profile文件最后增加一行，和自己路径保持一致
export PATH=$PATH:/home/lqf/webrtc/google_appengine
# 生效
source /etc/profile
```

## 3.5 安装go

安装go

```
sudo apt-get install golang-go
#检测go 版本
go version
#go version go1.6.2 linux/amd64
```

创建go工作目录

```
#创建go工作目录
mkdir -p ~/webrtc/goworkspace/src
```

```
#配置环境变量：在/etc/profile文件最后增加一行：
sudo vim /etc/profile
```

添加

```
export GOPATH=/home/lqf/webrtc/goworkspace
# 然后执行
source /etc/profile
```

## 3.6 安装apprtc

```
cd ~/webrtc/
# 先尝试使用github的下载连接
git clone https://github.com/webrtc/apprtc.git
# 如果github的连接下不了就用码云的链接
git clone https://gitee.com/sabergithub/apprtc.git
#将collider的源码软连接到go的工作目录下
ln -s /home/lqf/webrtc/apprtc/src/collider/collider $GOPATH/src
ln -s /home/lqf/webrtc/apprtc/src/collider/collidermain $GOPATH/src
ln -s /home/lqf/webrtc/apprtc/src/collider/collidertest $GOPATH/src

#下一步在编译 go get collidermain: 被墙
#报错: package golang.org/x/net/websocket: unrecognized import path "golang.org/x/net/websocket"
#所以先执行:
mkdir -p $GOPATH/src/golang.org/x/
cd $GOPATH/src/golang.org/x/
git clone https://github.com/golang/net.git net
go install net

#编译collidermain
cd ~/webrtc/goworkspace/src
go get collidermain
go install collidermain

```

## 3.7 安装coturn

```
sudo apt-get install libssl-dev
sudo apt-get install libevent-dev

# 返回webrtc目录
cd ~/webrtc
#git clone https://github.com/coturn/coturn
#cd coturn
# 提供另一种安装方式turnserver是coturn的升级版本
wget http://coturn.net/turnserver/v4.5.0.7/turnserver-4.5.0.7.tar.gz
tar xfz turnserver-4.5.0.7.tar.gz
cd turnserver-4.5.0.7

./configure
make
sudo make install
```

## 3.8 安装Nginx

注意安装的时候要带ssl

```
sudo apt-get update
#安装依赖：gcc、g++依赖库
sudo apt-get install build-essential libtool
#安装 pcre依赖库（http://www.pcre.org/）
sudo apt-get install libpcre3 libpcre3-dev
#安装 zlib依赖库（http://www.zlib.net）
sudo apt-get install zlib1g-dev
#安装ssl依赖库
sudo apt-get install openssl


#下载nginx 1.15.8版本
wget http://nginx.org/download/nginx-1.15.8.tar.gz
```

```
tar xvzf nginx-1.15.8.tar.gz
cd nginx-1.15.8/


# 配置，一定要支持https
./configure --with-http_ssl_module

# 编译
make

#安装
sudo make install
```

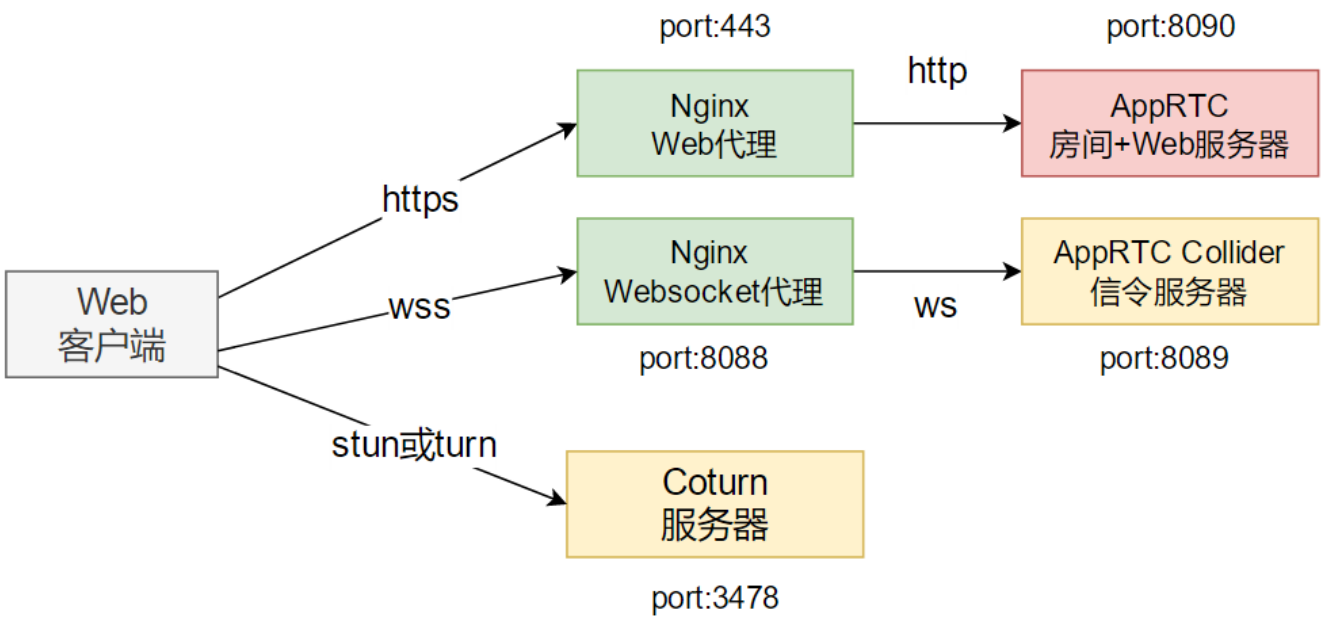
默认安装目录：/usr/local/nginx

启动：sudo /usr/local/nginx/sbin/nginx

停止：sudo /usr/local/nginx/sbin/nginx -s stop

重新加载配置文件：sudo /usr/local/nginx/sbin/nginx -s reload

# 4 配置与运行



# 4.1 coturn 打洞+中继服务器

配置防火墙，允许访问3478端口（含tcp和udp，此端口用于nat穿透）

可以前台执行：sudo turnserver -L 0.0.0.0 -a -u lqf:123456 -v -f -r nort.gov 然后退出，再到后台去执行

```
sudo nohup turnserver -L 0.0.0.0 -a -u lqf:123456 -v -f -r nort.gov &
#账号 lqf 密码:123456 这一步随便给，但是后面配置apprtc时需要用到
#命令后加 & ,执行起来后按 ctr+c,不会停止
```

```
#开启新窗口 执行
lsof -i:3478
#输出大致这样的成功
COMMAND     PID USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
turnserve 30299 root   16u  IPv4  61868      0t0  UDP *:3478
turnserve 30299 root   17u  IPv4  61869      0t0  UDP *:3478
turnserve 30299 root   32u  IPv4  61879      0t0  TCP *:3478 (LISTEN)
turnserve 30299 root   33u  IPv4  61880      0t0  TCP *:3478 (LISTEN)
```

# 4.2 collider 信令服务器

配置防火墙，允许访问8089端口（tcp，用于客户端和collider建立websocket信令通信）

```
# 执行collider 信令服务器
sudo nohup $GOPATH/bin/collidermain -port=8089 -tls=false  -room-
server="http://192.168.2.143:8090" &
```

-room-server="http://192.168.2.143:8090" 实际是连接房间服务器的地址

```
#同样检查是否成功
sudo lsof -i:8089
#显示
```

```
COMMAND     PID USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
colliderm 30354 root    3u  IPv6  62696      0t0  TCP *:8089 (LISTEN)
```

# 4.3 apprtc 房间服务器

先安装python的request模块

## 4.3.1 安装pip

下载setup-python工具

```
cd /home/lqf/webrtc
wget https://pypi.python.org/packages/2.7/s/setuptools/setuptools-0.6c11-py2.7.egg  --no-check-certificate
chmod +x setuptools-0.6c11-py2.7.egg
sudo ./setuptools-0.6c11-py2.7.egg
wget https://pypi.python.org/packages/source/p/pip/pip-1.5.4.tar.gz
tar -xf pip-1.5.4.tar.gz
cd pip-1.5.4/
sudo python setup.py install
sudo pip install requests
```

## 4.3.2 修改配置文件

配置防火墙，允许访问8090端口（tcp，此端口用于web访问）

配置文件修改（主要是配置apprtc对应的conturn和collider相关参数）

```
 vim /home/lqf/webrtc/apprtc/src/app_engine/constants.py
```

修改后（填的都是外网IP，为了适合更多数朋友测试，我这里用的是内网的环境，在公网部署填入公网IP即可）

# ICE_SERVER_OVERRIDE = None 注释掉

```
```

```
# Turn/Stun server override. This allows AppRTC to connect to turn servers
# directly rather than retrieving them from an ICE server provider.
# ICE_SERVER_OVERRIDE = None
# Enable by uncomment below and comment out above, then specify turn and stun
ICE_SERVER_OVERRIDE  = [
  {
     "urls": [
       "turn:192.168.2.143:3478?transport=udp",
       "turn:192.168.2.143:3478?transport=tcp"
     ],
     "username": "lqf",
     "credential": "123456"
   },
   {
     "urls": [
       "stun:192.168.2.143:3478"
     ]
   }
 ]

ICE_SERVER_BASE_URL = 'https:192.168.2.143'
ICE_SERVER_URL_TEMPLATE = '%s/v1alpha/iceconfig?key=%s'
ICE_SERVER_API_KEY = os.environ.get('ICE_SERVER_API_KEY')

# Dictionary keys in the collider instance info constant.
WSS_INSTANCE_HOST_KEY = '192.168.2.143:8088'
WSS_INSTANCE_NAME_KEY = 'vm_name'
WSS_INSTANCE_ZONE_KEY = 'zone'
WSS_INSTANCES = [{
    WSS_INSTANCE_HOST_KEY: '192.168.2.143:8088',
    WSS_INSTANCE_NAME_KEY: 'wsserver-std',
    WSS_INSTANCE_ZONE_KEY: 'us-central1-a'
}]

```

进到apprtc目录

```
#编译
```

```
cd /home/lqf/webrtc/apprtc
sudo cnpm install
sudo grunt build
```

启动:

sudo /home/lqf/webrtc/google_appengine/dev_appserver.py --host=0.0.0.0 --port=8090 /home/lqf/webrtc/apprtc/out/app_engine --skip_sdk_update_check

```
# 默认端口是8080， 可以自己指定端口，我们这里指定为8090
sudo nohup  /home/lqf/webrtc/google_appengine/dev_appserver.py --host=0.0.0.0 --port=8090 /home/lqf/webrtc/apprtc/out/app_engine --skip_sdk_update_check &
#如果提示更新选择: n
```

此时可以通过谷歌浏览器访问测试:
http://192.168.2.143:8090/

```
#检查
sudo lsof -i:8090
#输出下列内容

```

# 4.4 nginx代理

## 4.4.1 生成证书

```
mkdir -p ~/cert
cd ~/cert
# CA私钥
openssl genrsa -out key.pem 2048
# 自签名证书
openssl req -new -x509 -key key.pem -out cert.pem -days 1095
```

## 4.4.2 Web代理

反向代理apprtc，使之支持https访问，如果http直接访问apprtc，则客户端无法启动视频音频采集（必须得用https访问)

**配置web服务器**

（1）配置自己的证书

ssl_certificate /home/lqf/cert/cert.pem;  // 注意证书所在的路径

ssl_certificate_key /home/lqf/cert/key.pem;

（2）配置主机域名或者主机IP

server_name 192.168.2.143;

完整配置文件：/usr/local/nginx/conf/conf.d/apprtc-https-proxy.conf

```
upstream roomserver {
   server 192.168.2.143:8090;
}
server {
    listen 443 ssl;
    ssl_certificate /home/lqf/cert/cert.pem;
    ssl_certificate_key /home/lqf/cert/key.pem;
    charset utf-8;
    # ip地址或者域名
    server_name 192.168.221.134;
    location / {
        # 转向代理的地址
        proxy_pass http://roomserver$request_uri;
        proxy_set_header Host $host;
    }
}
```

编辑nginx.conf文件，在末尾}之前添加包含文件

```
  include /usr/local/nginx/conf/conf.d/*.conf;
}
```

### 4.4.3 配置websocket代理

ws 不安全的连接 类似http

wss是安全的连接，类似https

完整配置文件：/usr/local/nginx/conf/conf.d/apprtc-websocket-proxy.conf

```
map $http_upgrade $connection_upgrade {
    default upgrade;
    '' close;
}
upstream websocket {
    server 192.168.2.143:8089;
}

server {
    listen 8088 ssl;
    ssl_certificate /home/lqf/cert/cert.pem;
    ssl_certificate_key /home/lqf/cert/key.pem;

    server_name 192.168.2.143;
    location /ws {
        proxy_pass http://websocket;
        proxy_http_version 1.1;
        proxy_connect_timeout 4s; #配置点1
        proxy_read_timeout 6000s; #配置点2，如果没效，可以考虑这个时间配置长一点
        proxy_send_timeout 6000s; #配置点3
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection $connection_upgrade;
    }
}
```

## 4.5 解决跨域问题

浏览器通话跨域问题 :pushState

Messages:Failed to start signaling: Failed to execute 'pushState' on 'History'

**解决方法**

修改文件apprtc/src/web_app/js/appcontroller.js

```
vim /home/lqf/webrtc/apprtc/src/web_app/js/appcontroller.js

#搜索  displaySharingInfo_ 大概是445行displaySharingInfo_函数第一行添加

roomLink=roomLink.replace("http","https");
```

最终结果（大概446行的修改）

```
AppController.prototype.displaySharingInfo_ = function(roomId,
roomLink) {
  roomLink=roomLink.replace("http","https");
  this.roomLinkHref_.href = roomLink;
  this.roomLinkHref_.text = roomLink;
  this.roomLink_ = roomLink;
  this.pushCallNavigation_(roomId, roomLink);
  this.activate_(this.sharingDiv_);
};
```

然后重新build

```
cd ~/webrtc/apprtc
sudo grunt build
```

在重新在前台运行

```
sudo  /home/lqf/webrtc/google_appengine/dev_appserver.py --
host=0.0.0.0 --port=8090 /home/lqf/webrtc/apprtc/out/app_engine --
skip_sdk_update_check
```

# 总结

（1）目录/home/lqf/webrtc

（2）IP： 192.168.2.143

（3）前后台执行的问题

（4）防火墙开发端口的问题

（5）端口规划的问题